A Systematic Approach for Diagnosing Multiple Delay Faults

Jayabrata Ghosh Dastidar and Nur A. Touba

Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712-1084
E-mail: {dghosh,touba}@ece.utexas.edu

Abstract

In the presence of multiple delay faults, automated diagnostic procedures that make a single fault assumption may give an incorrect diagnosis. In this paper, a systematic approach is proposed for delay fault diagnosis under a multiple fault assumption. Information from the failing test vectors are used to construct a list of single and multiple fault suspects that may have caused all of the observed faulty response. The list of suspects is then pruned and ranked in a novel way by using information from the passing test vectors combined with static timing information.

1. Introduction

A digital logic circuit must meet both functional as well as timing specifications. Delay testing is required to verify the temporal correctness of manufactured circuits. Two fault models are commonly used for timing defects: the gate delay fault model [1], [2], and the path delay fault model [3]-[5]. With the advent of deep submicron technology, delay faults are becoming more prevalent. With ever increasing clock frequencies, small delay defects that were previously harmless, are now starting to cause timing failures. Not only is delay fault testing needed but also delay fault diagnosis to improve yield and quality of integrated circuits. Some direct probing mechanisms exist (e.g., E-Beam probing), but their effectiveness is limited by such factors as multiple layers of metals, CMP (chemical mechanical polishing), long test lengths, and new package types like flip-chip. Most importantly, with shrinking device sizes, the search space for any direct probing technique has increased tremendously. Automated tools are needed for failure analysis to significantly prune down the search space for direct probing.

2. Previous Work

Most of the work in diagnosis has been based on the classical single stuck-at fault model. Fault dictionaries have commonly been used for fault diagnosis [6]. Test generation techniques have also been developed to generate test vectors to distinguish pairs of faults [7]-[9]. De and Gunda [10] proposed a scheme to handle multiple stuck-at faults by ranking the faults according to the likelihood of being present in the defective part.

Delay fault diagnosis is significantly more difficult than stuck-at fault diagnosis as a delay fault model has to consider the size of delay defects and is often harder to define. Cox and Rajski [8] have proposed a method that uses a sixteen-valued algebra for both stuck-at fault and delay fault diagnosis. Girard, *et al.* [12], have developed an efficient procedure based on critical path tracing [11] from a six-valued simulation. The advantage of the method in [12] is that it requires no delay size based fault models and only considers the fault-free circuit. The drawback is that it assumes a single point fault.

Here we briefly describe the scheme presented in [12]. Any line that is a potential cause of an observed incorrect output value will be termed a *suspect*. For each failing test vector pair, a 6-valued simulation is done to ascertain the signal values on each line. From each primary output having an incorrect value, critical path tracing is done to identify the suspects (i.e., the critical lines) that may have caused the incorrect value. Heuristics are used to reduce the number of critical paths traced. Then the set intersection of the suspect lists for all failing vectors is computed. This is the set of suspects that could have caused all the observed faulty behavior. Note that this method can be invalidated in the presence of multiple faults as the set intersection of the suspect lists can eliminate the actual faults from the suspect list. In deep submicron technology, where even small delay defects can cause timing problems, multiple point faults are a more realistic assumption. Also, for a manufacturing process that has not fully matured, a multiple point fault model will be more valuable to diagnose the defects to improve the quality of the process.

We propose a method for delay fault diagnosis under a multiple fault assumption. In our proposed method we not only derive information from the failing vectors to identify the possible suspects, but also use the passing vectors in a systematic way to further refine the set of suspects. We also use a novel approach based on static timing information to rank the faults according to their likelihood of being present in the defective part.

3. Proposed Method

Diagnosis under a multiple fault assumption is much more complex as many of the simplifying assumptions for single faults no longer hold good. In our proposed method, we develop a systematic approach to tackle this problem. Our method is capable of handling a multiple number of faults up to a certain extent. We rank the faults according to their likelihood of being the cause of the observed behavior of the circuit being diagnosed. Here we describe our method which consists of two main steps: suspect list generation and suspect list refinement.

3.1 Generating the Suspect List

For a test sequence, we collect all the test vector pairs for which the circuit-under-test has failed and associate with each pair the set of outputs that have registered a faulty value for that pair. We also collect the test vector pairs for which the circuit has produced no errors. The circuit is initially preprocessed to obtain a levelized order. We then use a 6-valued algebra for our simulation purposes based on the H6 algebra [13] as is used in [12]. The symbols used are the following: *S0* for static zero, *S1* for static one, *R1* for a rising transition, *F0* for a falling transition, *X0* for static-0 hazard, and *X1* for a static-1 hazard. The advantage of using this 6-valued algebra is that it does not depend on any gate propagation delay or delay fault size.

A pair of vectors is required to detect a delay fault. We take a vector pair $\{v_1, v_2\}$ from the set of collected vector pairs for which the circuit failed. 6-valued simulation is done for v_1 with all the inputs either *S0* or *S1*. Then 6-valued simulation is done for v_2 to obtain the signal values on each line. We already have the information about which outputs have registered a faulty value for this test vector pair. From each such primary output we do critical path tracing. For each failing vector pair v, we form a set $C_{v,j}$ of suspects (i.e. critical lines) obtained by critical path tracing from each failing output j. For all the other outputs that have fault-free values, we do not carry out critical path tracing. We define a set of prime suspects, *PS*, which is computed as the intersection of all suspects $C_{v,j}$, (i.e., $PS = \Pi\, C_{v,j}$ for all v and j). It is evident that the set PS contains all the suspects whose presence alone can explain all the failing primary outputs for each vector pair v that failed. If the set PS is empty, we can immediately conclude the presence of multiple faults. It is important to note here that if PS is not empty we cannot eliminate the possibility of multiple faults.

We also define a set MS, whose members are the suspects for multiple faults. MS is calculated by the set union of all $C_{v,j}$ minus PS, (i.e., $MS = \cup\, C_{v,j} - PS$ for all v and j). We also associate with each element in PS or MS the primary outputs that might have failed because of it. We repeat the above steps for all the failing vectors. If the number of suspect lines in PS and MS are small enough for our purposes we stop. But normally that is not expected to happen, so we need to refine this scheme to further eliminate suspects.

In the example in Fig. 1, gates $G1$ and $G3$ have delay faults. The primary outputs O_1 and O_2 have faulty values on application of the test vector pair *(01001,11000)*. Using critical path tracing starting from O_1, we have $C_1 = \{O_1, L_3, I_1\}$. Critical path tracing starting from O_2 results in $C_2 = \{O_2, L_{10}, I_5\}$. So the intersection of sets C_1 and C_2 gives $PS = \{\}$. The set union of C_1 and C_2 gives $MS = \{O_1, L_3, I_1, O_2, L_{10}, I_5\}$. Note that MS contains the actual faulty lines L_3 and L_{10} whereas PS is empty signifying the presence of multiple faults.

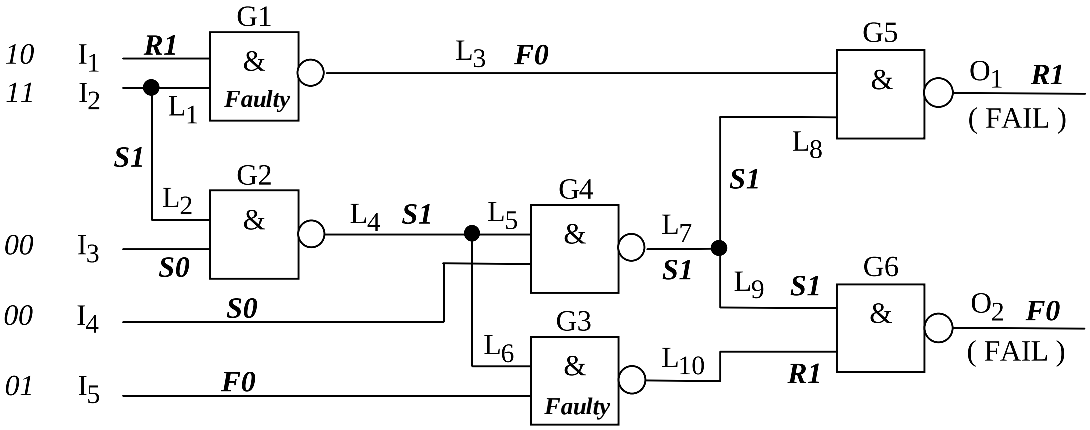

Figure 1. Signal values at Different Lines after Application of Vector Pair *(01001,11000)*
Faulty Gates: G1 and G3
$C_1 = \{O_1, L_3, I_1\}$; $C_2 = \{O_2, L_{10}, I_5\}$; $PS = \{\}$; $MS = \{O_1, L_3, I_1, O_2, L_{10}, I_5\}$

3.2 Refining the Suspect List

The refinement we propose is to make use of the vectors that have not produced any faulty output values. These vectors are already available as part of the test sequence applied on the circuit under test. So we can use them to refine our scheme without having to spend any effort on their generation or application to the circuit. In our scheme, we obtain information about the possible likelihood of a suspect causing an observed behavior without explicitly enumerating all possible paths passing through a suspect.

There can be a large number of paths passing through a suspect. Among the paths passing through a node, all of the paths, some of the paths, or none of the paths could have been robustly tested during the test sequence. If the normal delay through a path, added with the delay defect size (amount of delay over the specified delay) is longer than the clock period, then a faulty output will be observed when the path is robustly tested. So depending on the delay defect size, a suspect can cause a delay fault along all paths, some paths, or none of the paths passing through it. The possible delay defect size can range from zero to gross defect sizes. Our approach examines what delay defect size for a suspect can explain the observed behavior during the test sequence. For each suspect in the suspect list, we obtain two bounds: a lower bound and an upper bound whenever possible. The lower bound is the minimum possible delay defect size at a suspect that can explain the observed faulty behavior. The upper bound is the maximum possible delay defect size at a suspect that, when added to the normal delay of a path passing through the suspect, can explain the fault-free behavior when that path has been robustly tested. If for all the test vectors in the test sequence there exists no path through a suspect which, when robustly tested, has produced a fault-free output, then no upper bound is obtained. Suspects with no upper bound are ranked at the top of the suspect list and are ranked by the ascending order of their lower bounds. We now discuss the necessary steps to obtain the lower and upper bounds.

Looking at the paths that are robustly tested by a test vector pair, we obtain the upper bound for a suspect by finding the shortest path, in terms of delay, from a primary input to the suspect, and the longest path from the suspect to a fault-free primary output. The shortest path to the suspect gives the earliest arrival time for a transition at the suspect, and the longest path gives the most stringent timing requirement that was satisfied (since the observed output was fault-free). So the difference between the clock period and the sum of these two times is the largest possible delay at the suspect that would still be consistent with the observed fault-free response (i.e., the largest delay at the suspect that has been tested to be fault-free).

Looking at the paths that are tested by a two-pattern test, we obtain the lower bound for a suspect by finding the longest path, in terms of delay, from a primary input to the suspect, and the shortest path from the suspect to a faulty primary output. The longest path to the suspect gives the latest arrival time for a transition at the suspect, and the shortest path gives the least stringent timing requirement that was not satisfied (since the observed output was faulty). So the difference between the clock period and the sum of these two times is the smallest possible delay at the suspect that would still be consistent with the observed faulty response (i.e., the smallest delay at the suspect that has been tested to be faulty).

When obtaining the upper and lower bounds on the delay defect size, we use the specified min-max delays for each gate. We are always conservative in choosing the delay value. When considering the fault-free paths, we assume the minimum delay for each gate (i.e., assume the largest timing slack). When considering the faulty paths, we assume the maximum delay for each gate (i.e., assume the smallest timing slack).

If the smallest possible delay defect for a suspect that could have caused an observed faulty output (i.e., the lower bound), is greater than the largest delay defect for which a suspect has been robustly tested and observed to be fault-free (i.e., the upper bound), we eliminate that suspect from the suspect list. Otherwise we rank it according to the relative delay values in the upper and lower bounds. At the end of this procedure, we have a ranked list of final suspects. The suspects at the top of the list are the prime candidates for causing the observed behavior of the circuit under test.

4. Experimental Results

Delay fault diagnosis is heavily dependent on the defect size and the test set used. Here some experimental results are shown for a test set that was generated by the tool *Soprano* [14]. Table 1 summarizes the results obtained for the ISCAS 85 [15] benchmark circuits C432 and C880. Random faults of varying size were injected. The propagation delay for 2-input AND and OR gates was 2 units, and for inverters it was 1 unit. In the second column in Table 1, the fault locations are shown with their corresponding defect sizes in the third column. The fourth column shows the initial number of suspects before any elimination or ranking has been done. The fifth column shows the number of suspects after all the suspects having lower bound greater than upper bound have been eliminated. The last column shows the individual rank of each suspect in the final suspect list. As can be seen, in most cases the elimination and ranking are able to significantly narrow the search space. Note that this improvement comes at no cost in terms of addition test generation and test application. By simply extracting additional information from the passing tests, the diagnosis is improved.

Table 1. Experimental Results for Fault Diagnosis of Multiple Faults

Circuit	Fault Location	Defect Size	Initial Suspects	Suspects after Elimination	Rank
C432	Gate: [438] Gate: 171	2 units	25	18	8 15
C432	Gate: 63 Gate: 82	3 units	23	16	13 16
C432	Gate: 174 Gate: 63	3 units	23	16	11 15
C880	Gate: [379] Gate: 851	3 units	127	111	10 30
C880	Gate: 632 Gate: 541	3 units	110	95	18 20
C880	Gate: 466 Gate: 543	4 units	98	89	6 61
C880	Gate: 830 Gate: 146	5 units	37	36	2 21

5. Summary and Conclusions

In the presence of multiple delay faults, automated diagnostic procedures that make a single fault assumption may give an incorrect diagnosis. In this paper, a systematic approach has been proposed for delay fault diagnosis under a multiple fault assumption. Information from the failing test vectors is used to construct a list of single and multiple fault suspects that may have caused the faulty responses. The list of suspects is then pruned and ranked in a novel way by using information from the passing test vectors combined with static timing information. The end result is a suspect list that can be used to guide physical probing of the part.

The approach described here also provides a framework for improving the multiple delay fault diagnosis. If further diagnostic resolution is required, additional test generation can be performed to eliminate more suspects by tightening the upper and lower bounds. This narrows down the list of suspects and ranks them more appropriately thereby reducing the search space for hard-to-diagnose multiple delay faults.

Acknowledgements

This material is based on work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. DABT63-97-C-0024, and in part by the National Science Foundation under Grant No. MIP-9702236.

References

[1] Wagner, K.D., “The error latency of delay faults in combinational and sequential circuits”, *Proc. International Test Conference,* pp. 334-341, Nov. 1985.

[2] Waicukauski, J.A., E. Lindbloom, B. Rosen, and V. Iyenger, “Transition fault simulation”, *IEEE Design & Test,* vol. 4, pp. 32-38, Apr. 1987.

[3] Lesser, J.P., and J. J. Shedletsky, “An experiment delay test generator for LSI logic”, *IEEE Trans. Computers*, vol. 29 pp. 235-248, Mar. 1980.

[4] Smith, G.L., “Model for delay faults based upon paths”, *Proc. International Test Conference*, pp. 342-349, 1985.

[5] Lin, C.J., and S. M. Reddy, “On delay fault testing in logic circuits”, *IEEE Trans. Computer- Aided Design,* vol. CAD-6, pp. 694-703, Sept. 1987.

[6] Abromovici, M., M. A. Breuer, and A. D. Friedman, *Digital System Testing and Testable Design,* Computer Science Press, New York, 1990.

[7] Camurati, P., D. Medina, P. Prineto, and M. S. Reorda, “A Diagnostic Test Pattern Generation”, *Proc. International Test Conference,* pp. 52-58, 1990.

[8] Cox, H., and J. Rajski, “ A Method of Fault Analysis for Test Generation and Fault Diagnosis ”, *IEEE Trans. Computer-Aided Design,* vol. 7, no. 7, pp. 813-833, 1998.

[9] Gruning, T., U. Mahlstedt, and H. Koopmeiners, “DIATEST: A Fast Diagnostic Test Pattern Generator for Combinational Circuits”, *Proc. International Conference on Computer-Aided Design (ICCAD)*, pp. 194-197, 1991.

[10] De, K., and A. Gunda, “ Failure Analysis for Full-Scan Circuits,” *Proc. International Test Conference*, pp. 636-645, 1995.

[11] Abromovici, M., P. R. Menon, and D. T. Miller, “Critical Path Tracing - An Alternative to Fault Simulation”, *Proc. 20th Design Automation Conference*, pp. 214-220, 1983.

[12] Girard, P., C. Landrault, S. Pravossoudovitch, “A Novel Approach to Delay-Fault Diagnosis”, *Proc. 29th Design Automation Conference*, pp. 357-360, 1992.

[13] Hayes, J.P., “Digital Simulation with Multiple Logic Values”, *IEEE Trans. Computer-Aided Design*, vol. 5, no. 2, pp. 274 -283, April 1986.

[14] Lee, H.K., and D.S. Ha, “SOPRANO: An Efficient Automatic Test Pattern Generator for Stuck-Open Faults in CMOS Combinational Circuits," *Proc. 27th Design Automation Conference*, pp. 660-666, 1990.

[15] Brglez, F., and H. Fujiwara, “A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortan,” *Proc. of Int. Symp. Circuits and Systems*, pp. 663-698, 1985.